Common

-shell

- You must not use the “system()” function. Instead, use the “exec()” family.
- The shell runs in an **infinite loop**, reading one command line at a time.
- While executing a command in a child process, the **parent process must wait** until the command finishes.
  - You can test this behavior using the “sleep” command.
- **No auto-completion or suggestions** are required.
- You **do not need to implement background execution** of processes.
- When the mini shell exits, the **parent process must wait** for all child processes to finish before terminating itself.

-input

- Each command consists of **at most 20 words (tokens)**.
- You may assume test inputs contain **no empty commands and no unknown commands.**
- You do **not need to support shell metacharacters** or **features** except for the **pipe (|).**
  - Except for the **single quotes used in alias** definitions, **no single or double quotes will appear in the input commands**.
- Each command appears on a single line.
- All test cases end with a line containing “quit₩n”.
- **No test case contains more than 10 commands**, including the final “quit₩n”.
- Inputs will **not have leading or trailing spaces or tabs**.

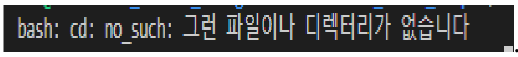
-output

- Ignore any literal prompt character like $ (i.e., prompts are not part of the graded output).

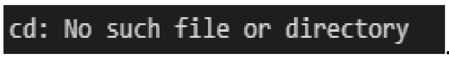
External commands

- You are not required to implement external commands; **simply execute them using exec*()**.
- For grading, all external commands will be located in /usr/bin or /bin.
  - E.g., /bin/ls, /bin/cat, /bin/grep, /bin/wc.
- You **must accept options to external commands.**
  - E.g., ls -a.
- For external commands (e.g., /bin/cat reading a non-existent file), do not intercept errors—let the program's own stderr output appear directly.

cd commands (Built-in)

- **Handle** "cd ~" by changing to $HOME.
  - You must handle exactly "cd ~". Handling "cd ~/[additional path]" is **not required**.
- You **do not have to handle** the case "cd" with no argument.
- For cd failures, print the message derived from errno to stderr
  - E.g., Use perror("cd");
  - **Note**: In a real bash shell, the error message may appear with a prefix or in the system's locale(Korean) such as 

    ```
    bash: cd: no_such: 그런 파일이나 디렉터리가 없습니다
    ```

    .
    But in your mini_shell, it may appear as 

    ```
    cd: No such file or directory
    ```

    .
    Either form is acceptable as long as the error message is derived from errno and printed to stderr.
- All directory names used in grading contain only English letters and digits, and no spaces.

alias commands

- The maximum number of **unique alias names** is **20**.
- The aliased command (right-hand side) must be enclosed in single quotes.
  - E.g., alias ll=ls –al (X), alias ll='ls –al' (O)
- If the same alias name is defined multiple times, **the most recent one takes effect.**
- Aliases may appear only as the **first token** of the entire line or the first token **after a pipe.**
  - At most one alias per line.
  - No recursive alias expansion.
  - E.g., alias my_wc ='wc -l', my_wc | grep a (O)
  - E.g., alias my_wc='wc -l', cat text.txt | my_wc (O)
  - E.g., alias list='abcdef', echo list (X)
  - E.g., alias list='abcdef', cat text.txt | grep list (X)
- There **is at most one aliased token per line**.
- Aliases **can accept additional options** at use time
  - E.g., alias my_wc='wc -c', "cat t | my_wc -m"
  - This is equivalent to "cat t | wc -c -m"
- Alias names will not duplicate existing built-in or external command names.
- You **do not need to consider examples** that print alias names using echo
  - E.g., alias my_ll='ls -al', echo ll (X)
  - Such cases will not be tested
- You do not need to implement **unalias**.
- You do not need to handle a line that is only the word "alias".
  - E.g., no listing required.

pipe("|") commands

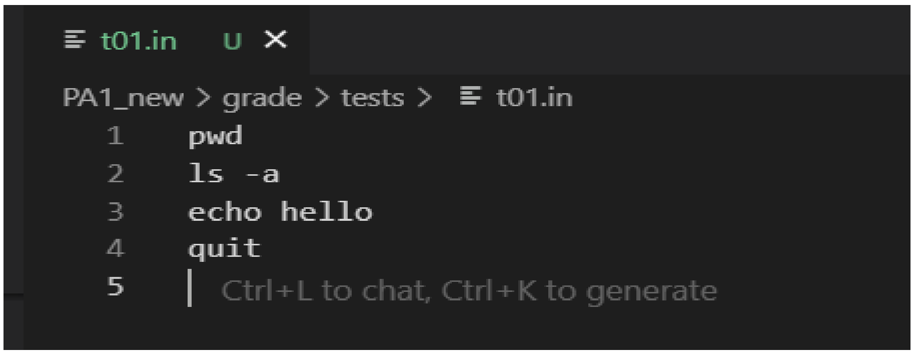
- For pipelines (up to 2 pipes == up to 3 processes), **the parent must wait for all child processes in the pipeline.**
- There must be **exactly one space** on both sides of "|"
  - "ls –al | grep A" (O)
  - "ls –al |grep A" (X)
  - "ls –al | grep A" (X)
- Maximum number of pipes per command line is 2
  - E.g., ls -al | grep A | grep B

Self test

The example test case files will be provided. However, their expected outputs will not be provided, since the results may vary depending on the environment.

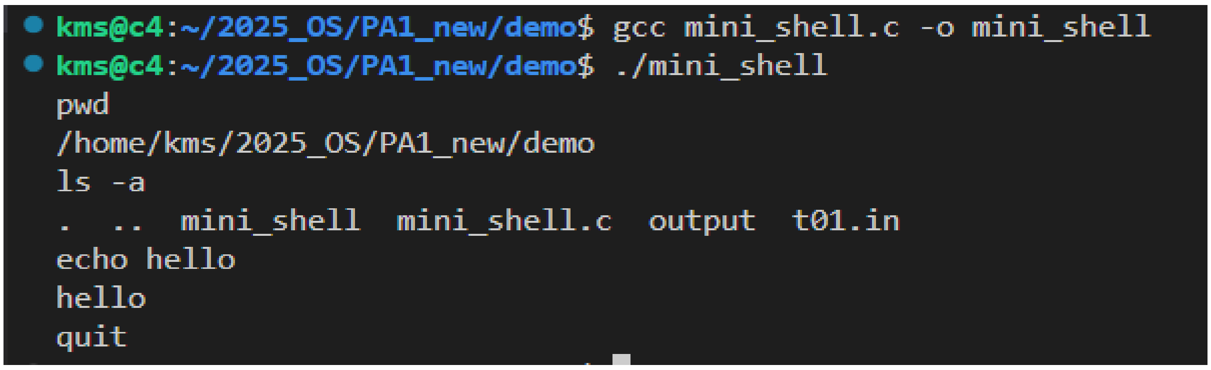
Instead, you should verify the correctness of your "mini_shell" by comparing its output with that of a real shell using the given test case files.

- Test input



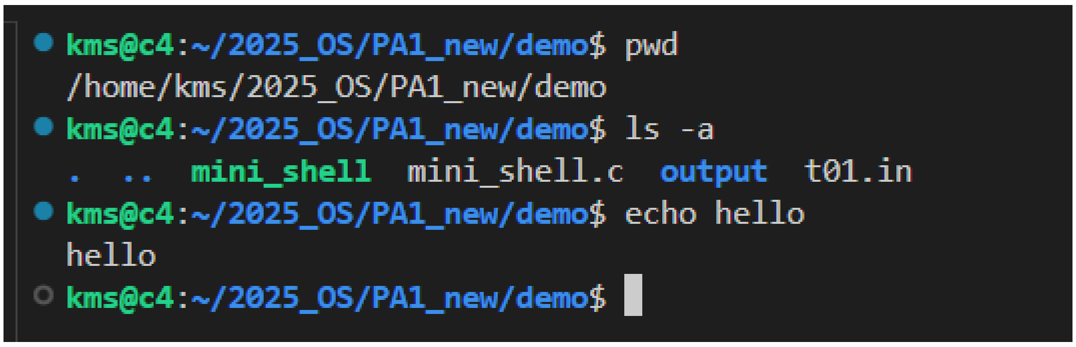
```
t01.in  U ×
PA1_new > grade > tests > t01.in
pwd
ls -a
echo hello
quit
|  Ctrl+L to chat, Ctrl+K to generate
```

- mini_shell



```
kms@c4:~/2025_OS/PA1_new/demo$ gcc mini_shell.c -o mini_shell
kms@c4:~/2025_OS/PA1_new/demo$ ./mini_shell
pwd
/home/kms/2025_OS/PA1_new/demo
ls -a
.  ..  mini_shell  mini_shell.c  output  t01.in
echo hello
hello
quit
```

- Real shell

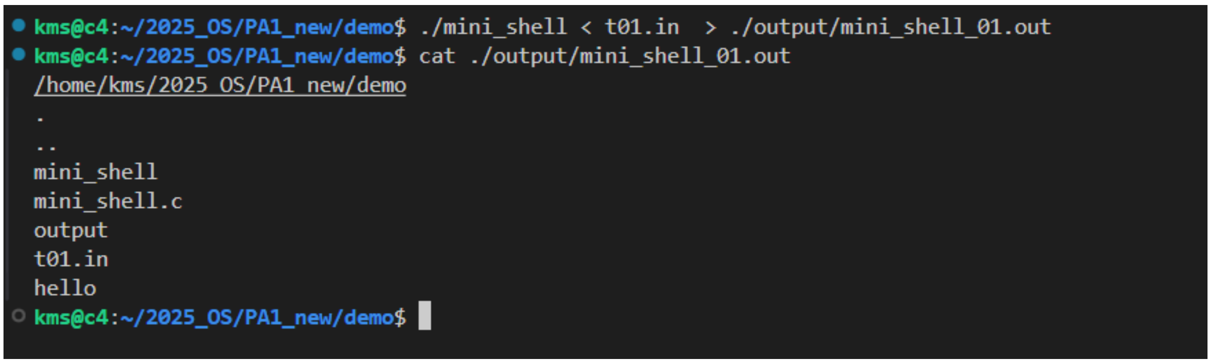


```
kms@c4:~/2025_OS/PA1_new/demo$ pwd
/home/kms/2025_OS/PA1_new/demo
kms@c4:~/2025_OS/PA1_new/demo$ ls -a
.  ..  mini_shell  mini_shell.c  output  t01.in
kms@c4:~/2025_OS/PA1_new/demo$ echo hello
hello
kms@c4:~/2025_OS/PA1_new/demo$ 
```

Note that the real shell does not support a command like "quit".

Or, you can compare results using file redirection

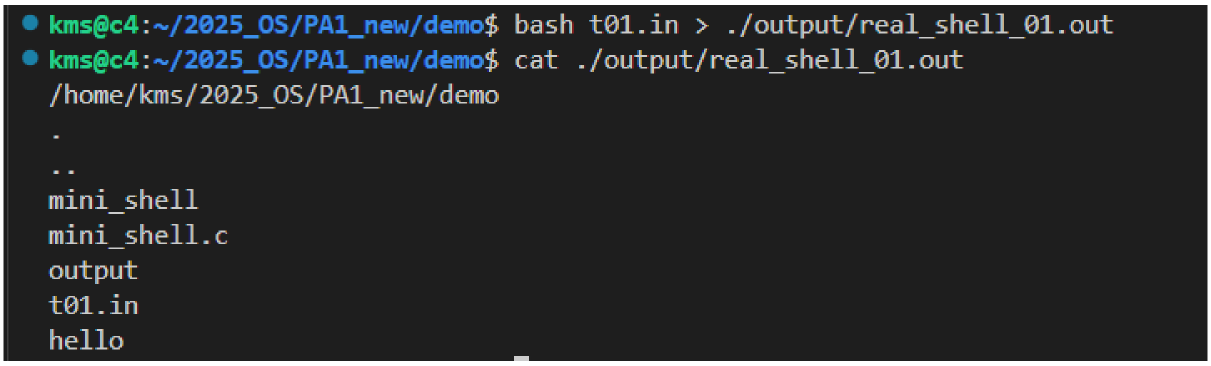
- mini_shell



```
kms@c4:~/2025_OS/PA1_new/demo$ ./mini_shell < t01.in  > ./output/mini_shell_01.out
kms@c4:~/2025_OS/PA1_new/demo$ cat ./output/mini_shell_01.out
/home/kms/2025_OS/PA1_new/demo
.
..
mini_shell
mini_shell.c
output
t01.in
hello
kms@c4:~/2025_OS/PA1_new/demo$ 
```

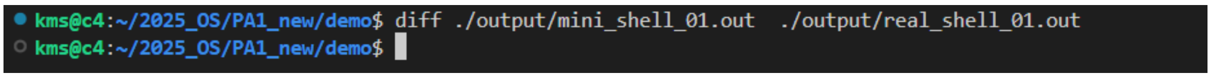
- real shell

(To obtain the result from the real shell, we recommend removing the quit command from the test case.)



```
kms@c4:~/2025_OS/PA1_new/demo$ bash t01.in > ./output/real_shell_01.out
kms@c4:~/2025_OS/PA1_new/demo$ cat ./output/real_shell_01.out
/home/kms/2025_OS/PA1_new/demo
.
..
mini_shell
mini_shell.c
output
t01.in
hello
```

- Diff



```
kms@c4:~/2025_OS/PA1_new/demo$ diff ./output/mini_shell_01.out  ./output/real_shell_01.out
kms@c4:~/2025_OS/PA1_new/demo$ 
```

If the contents of the two output files are the same, the diff command will produce no output.

Submission

- Submit your single mini_shell source code named [studentID].c
    - We will only accept submissions in the form of a **.c** file, which will be compiled and graded. Therefore, you must not implement the assignment in other languages such as **C++ (.cpp) or Python (.py)**.
    - We are going to compile your code using **gcc**.